Preprint ANL/MCS-P1748-0510

Noname manuscript No.
(will be inserted by the editor)

A preconditioning technique for Schur complement systems arising in stochastic optimization

Cosmin G. Petra · Mihai Anitescu

Received: date / Accepted: date

Abstract Deterministic sample average approximations of stochastic programming problems with recourse are suitable for a scenario-based parallelization. In this paper the parallelization is obtained by using an interior-point method and a Schur complement mechanism for the interior-point linear systems. However, the direct linear solves involving the dense Schur complement matrix are expensive, and adversely affect the scalability of this approach. We address this issue by proposing a stochastic preconditioner for the Schur complement matrix and by using Krylov iterative methods for the solution of the dense linear systems. The stochastic preconditioner is built based on a subset of existing scenarios and can be assembled and factorized on a separate process before the computation of the Schur complement matrix finishes on the remaining processes. The expensive factorization of the Schur complement is removed from the parallel execution flow and the scaling of the optimization solver is considerably improved with this approach. The spectral analysis indicates an exponentially fast convergence in probability of the eigenvalues of the preconditioned matrix with the number of scenarios incorporated in the preconditioner. Numerical experiments performed on the relaxation of a unit commitment problem show good performance, in terms of both the accuracy of the solution and the execution time.

Keywords stochastic programming · saddle-point preconditioning · Krylov methods · interior-point method · sample average approximations · parallel computing

1 Introduction

Stochastic programming (SP) is concerned with solving optimization problems involving uncertainty in the objective and/or the constraints. In this paper we consider two-stage stochastic problems with recourse having quadratic objective functions in each stage. Although these problems are nonlinear optimization problems [7], they are

C. Petra · M. Anitescu
Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA
E-mail: petra,anitescu@mcs.anl.gov

known as two-stage stochastic quadratic problems with recourse (TSQP) [41]. TSQPs are mathematically formulated as

$$\min \left(\frac{1}{2}x^T Qx + c^T x\right) + \mathcal{G}(x) \text{ subject to } Tx = b,\ x \geq 0, \tag{1}$$

where $\mathcal{G}(x) = \mathbb{E}[G(x, \xi(\omega))]$ and $G(x, \xi(\omega))$ is the optimal value of the second-stage problem

$$\min \frac{1}{2}u^T Q(\omega)u + c(\omega)^T u \text{ subject to } W(\omega)u = b(\omega) - T(\omega)x,\ u \geq 0. \tag{2}$$

Here $\xi(\omega) = (Q(\omega), c(\omega), T(\omega), W(\omega), b(\omega))$ denotes the data of the second-stage problem and is viewed as a random vector defined over a probability space Ω with a known probability distribution P. The expected value $\mathbb{E}[G(x, \xi(\omega))]$ is taken with respect to $\omega \in \Omega$. We also assume that Q and $Q(\omega)$ are symmetric positive semidefinite and that T, the technology matrices $T(\omega)$, and the recourse matrices $W(\omega)$ have full rank.

The first stage of the problem is the present moment of time, while the second stage corresponds to a future point in time at which the uncertainty is revealed. However, a decision x must be taken before the actual realizations of the uncertain parameters of the second stage become available. The optimal decision x also must take into account a "recourse" action for the possible outcome of the uncertainty. In the context of stochastic programming problems with recourse, the objective consists of not only the usual "operation cost" $\left(\frac{1}{2}x^T Qx + c^T x\right)$ but also the expected value $\mathbb{E}[G(x, \xi(\omega))]$ of the "recourse cost" taken with respect to all possible outcomes of the uncertainty.

In the case of a finitely supported distribution P, that is, there exist a finite number of realizations $\xi_1, \xi_2, \ldots, \xi_N$ of the random vector $\xi(\omega)$, the recourse function is a weighted average

$$\mathcal{G}(x) = \sum_{i=1}^{N} p_i G(x, \xi_i),$$

where p_i is the probability of ξ_i to occur, $i = 1, 2, \ldots, N$. Consequently, the SP problem (1)-(2) is a deterministic optimization problem with a special structure, see Section 2.1, but standard derivative-based methods cannot be applied because of the nonsmooth (piecewise differentiable, more exactly) recourse term. The reader may refer to [4,7] for a detailed survey of algorithms for SP problems. The L-shaped method [52] and its variants use Benders' decomposition of the primal problem or a Dantzig-Wolfe decomposition of the dual to exploit the special structure of the two-stage SP problem. Based on cutting-plane subgradient approximation of the objective, this class of methods builds a lower convex, nondifferentiable estimate of the recourse cost function and iterate by minimizing this estimate. Stabilization techniques such as quadratic regularization [50] or an l_∞ trust-region approach [35] are needed to make this class of methods robust. Among the alternatives to subgradient-based methods we mention Lagrangian-based approaches [47,34,48,49] and direct approaches that use simplex or an interior-point method to solve the equivalent deterministic convex quadratic programming problem of the form given by (7) (see Section 2.1).

The direct approach is competitive with other methods only if the optimization method takes into account the staircase structure of the constraints' Jacobian and the separability of the objective function of the deterministic problem. In this paper we use a primal-dual interior-point method (IPM) for solving the deterministic problem. This class of methods provides a unified framework for linear, quadratic and nonlinear continuous programming problems and requires virtually the same linear algebra. Based on Newton's method, IPMs exhibit polynomial complexity for convex programming problems and local superlinear or quadratic convergence for a large class of problems [53]. They also have emerged in the past twenty-five years as a practical method for solving large-scale optimization problems.

For SP problems having a continuous probability distribution, the evaluation of the recourse function $\mathcal{G}(x)$ requires integration of the function $G(x, \xi(\omega))$ in the space of the random parameters $\xi(\omega)$ [7]. Even if the integrand $G(x, \xi(\omega))$ is easily evaluated, the integration is usually untractable from a computational point of view even for a relatively small size of $\xi(\omega)$. On the other hand, when the probability space $\Omega = \{\omega_1, \omega_2, \ldots\}$ is discrete, the expectation term $\mathcal{G}(x)$ is a weighted average of $G(x, \xi(\omega_i)$, $i = 1, 2, \ldots$, and therefore the second stage has to be solved for each possible outcome of Ω, possibly infinitely many times if Ω is not finite. Even in the finite case, the cardinality of Ω can be extremely large since it increases exponentially with the number of independent random variables of $\xi(\omega)$. In order to deal with the issue of numerical integration for the continuous case and infinite or extremely large cardinality in the discrete case, sampling methods are used to obtain a discrete finite approximation of the randomness that is tractable from a computational point of view. Among the sampling techniques used in the context of stochastic programming, we mention Monte Carlo methods (covered in [7, Chapter 10] and [51, Chapter 5]), the Latin hypercube sampling method [38], and the importance sampling method [16].

Two fundamental approaches emerge from the interplay of optimization and sampling. The first is the "interior sampling" approach that performs the optimization directly on (1) and relies on sampling to approximate the recourse function and its derivative information. Methods following this approach include the stochastic quasi-gradient method [20], the stochastic Newton method [5], the stochastic decomposition method [32], and the L-shaped method based on importance sampling of [16]. Unfortunately, repeated samplings make this approach infeasible for applications in which the cost of the sampling is comparable with or larger than the cost of optimization. Weather-based simulation and control problems [14] are such examples.

In the second approach, "exterior sampling", the sampling is done only once at the beginning of the optimization process. A finite sample $\xi_1, \xi_2, \ldots, \xi_N$ of N realizations (scenarios) of the random vector $\xi(\omega)$ is used to approximate the expected value $\mathbb{E}[G(x, \xi(\omega))]$ by averaging the values $G(x, \xi_i)$, $i = 1, 2, \ldots, N$. The so-called sample average approximation (SAA) problem obtained with this approach is a convex quadratic deterministic programming problem having the form (7) with equiprobable scenarios $p_1 = \ldots = p_N = 1/N$. From a computational point of view, the SAA problem is a stochastic programming problem with a finite number of realizations. Therefore, any of the abovementioned numerical methods for the discrete SP problems can be applied to SAA SP problems.

SAA problems can become extremely large. For example in [24], an SAA problem having more than 1 billion variables was solved by using a primal-dual interior-point method. SAA SP problems of this size and even considerably smaller sizes can be handled only in a parallel environment with distributed memory, however, this requires the problem to be decomposed into subproblems that can be solved independently on different processors. In the context of subgradient-based or Lagrangian-based methods, the decomposition is obtained by evaluating the recourse function and obtaining derivative information independently for each scenario. Once an approximation of the recourse term is available, the optimization in the space of the first-stage variables is done in serial and thus causes a bottleneck in the parallel execution flow. On the other hand, the decomposition of the problem in the context of IPMs is usually achieved at the linear algebra level by taking advantage of the block-separable form of the objective function and the half-arrow shape of the Jacobian. This special structure allows most of the work related to IPM linear solves to be done independently for each scenario when a Schur complement mechanism is used. Decompositions of the SP problems based on the Schur complement and interior-point methods can be found as early as 1988 in the work of Birge and Qi [8]. Recently, IPM-based decomposition for SP problems was implemented in state-of-the-art software packages such as OOPS [25,26,28] and IPOPT [55]. A slightly different IPM decomposition is the log-barrier Benders-like decomposition introduced by Zhao in [58] for linear problems and extended by Mehrotra and Ozevin to the quadratic case [41] and to two-stage stochastic conic problems [40, 42]. A potential advantage of the log-barrier Benders decomposition consists of adaptively adding or removing scenarios during the optimization. However, the two IPM approaches share the same linear algebra, and the technique presented in this paper applies equally to both.

In the early 1990s, because of the limited network bandwidth, the IPM decomposition based on the Schur complement did not prove to be very scalable since it moves large matrices across processors. With the new high-speed interconnects, however, the network bandwidth does not have the same limiting effect, and the main drawback of IPM decomposition is similar to the one of L-shaped methods, namely, the linear algebra of the first stage cannot be started until all the second-stage information is available. More specifically, the only work that cannot be done independently for each scenario consists of linear solves with the Schur complement of the first-stage Hessian subblock in the entire Hessian matrix. The Schur complement for two-stage stochastic programming is a saddle-point linear system of the form

$$\begin{bmatrix} H & T^T \\ T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix},$$

where the block $H \in \mathbb{R}^{n_{x_0} \times n_{x_0}}$ is almost completely dense (n_{x_0} is the number of the first-stage variables). The presence of the dense block H incurs a cost of approximately $O(n_{x_0}^3)$ for each linear solve with the Schur complement matrix. For problems having a large number of first-stage variables, the $O(n_{x_0}^3)$ cost becomes dominant for a relatively large number of processors and adversely affects the scalability of the IPM decomposition method.

In this paper we propose an alternative method for the solution of the Schur complement linear systems. The goal of the current work is to remove the cost of the direct factorization of the Schur matrix associated with the first-stage variables from the parallel execution flow. We substitute the direct solve of the Schur complement matrix with Krylov subspace iterative methods. In the context of the ill-conditioned saddle-point linear systems arising from the primal-dual interior-point algorithms, Krylov methods are known to perform well only when used with a preconditioner. A popular preconditioning approach for the saddle-point linear systems is the constraint preconditioner [2,17,33,36,44] that replaces the $(1,1)$ block of the matrix by an easily invertible approximation matrix, while keeping the constraints blocks $(1,2)$ and $(2,1)$ intact. For example, diagonal approximations of the $(1,1)$ block have been used in [3], and implicit factorizations of the same block have been proposed in [18,19].

The *stochastic preconditioner* we propose in this paper approximates the $(1,1)$ block from the Schur complement matrix by incorporating only a subset of available scenarios of the SAA problem. Factorization of the preconditioner is carried out on a separate process *before all scenarios are finished* and the bottleneck is thus removed. We analyze the preconditioner from the perspective of the theory of large deviations and show that the eigenvalues of the preconditioned matrix cluster exponentially fast around 1 with the number of the scenarios incorporated in the preconditioner. In practice, the preconditioner allows the Krylov methods to solve the ill-conditioned IPM linear systems with the same accuracy as a direct solver in a small number of iterations.

The paper is organized as follows. In Section 2 we present the linear algebra required by interior-point methods for the solution of the SAA problem. The stochastic preconditioners are introduced and analyzed in Section 3, and the details of their implementation are presented in Section 4. In Section 5 we investigate and report on the practical performance of the preconditioners. The conclusions of this work and related future research directions are given in Section 6.

2 Linear algebra in interior-point methods

In this section we present the linear algebra operations needed by path-following interior-points methods to solve convex quadratic programming problems of the form

$$\min \frac{1}{2}x^T Qx + c^T x \text{ subject to } \ Ax = b,\ x \geq 0. \tag{3}$$

Path-following interior-point methods make use of the "central path", which is a continuous curve $(x(\mu), y(\mu), z(\mu))$, $\mu > 0$, satisfying

$$\begin{aligned} Qx - A^T y - z &= -c, \\ Ax &= b, \\ xz &= \mu e, \\ x, z &> 0. \end{aligned} \tag{4}$$

Here $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$ correspond to the Lagrange multipliers, $e = [1\ 1\ \ldots\ 1]^T \in \mathbb{R}^n$ and xz denotes the componentwise product.

In the case of a feasible problem (3), the above system has a unique solution $(x(\mu), y(\mu), z(\mu))$ for any $\mu > 0$, and, as μ approaches zero, $(x(\mu), y(\mu), z(\mu))$ approaches a maximal complementarity solution of (3), see Chapter 2 in [53]. A path-following method is an iterative numerical process that follows the central path in the direction of decreasing μ toward the solution set of the problem. The iterates generated by the method generally do not stay on the central path. Rather, they are located in a controlled neighborhood of the central path that is a subset of the positive orthant.

In the past two decades, predictor-corrector methods have emerged as practical path-following IPMs in solving linear and quadratic programming problems. Among the predictor-corrector methods, the most successful is Mehrotra's predictor-corrector algorithm. Although Mehrotra [39] presented his algorithm in the context of linear programming, it has been successfully applied also to convex quadratic programming [21] and standard monotone linear complementarity problems [56]. It also has been widely used in the implementation of several IPM-based optimization packages, including OB1 [37], HOPDM [23], PCx [15], LIPSOL [57], and OOQP [21].

Two linear systems of the form (5) are solved at each IPM iteration to obtain the predictor and corrector search directions. For a detailed description of Mehrotra's method used in this paper we refer the reader to [21,39,53]. Let us denote the kth IPM iteration by (x_k, y_k, z_k). Also let X_k and Z_k denote the diagonal matrices with the (positive) entries given by x_k and z_k. The linear system solved during both the predictor and corrector phase to obtain the seach direction $(\Delta x_k, \Delta y_k, \Delta z_k)$ is

$$\begin{aligned} Q\Delta x_k - A^T\Delta y_k - \Delta z_k &= r_k^1 \\ A\Delta x_k &= r_k^2 \\ Z_k\Delta x_k + X_k\Delta z_k &= r_k^3. \end{aligned} \tag{5}$$

While the right-hand sides r_k^1, r_k^2, and r_k^3 are different for the predictor and the corrector, the matrix remains the same (this feature gives important computational savings, since only one factorization, not two, per IPM iteration is required).

By performing block elimination for Δz_k, the linear systems (5) can be reduced to the following symmetric indefinite linear system

$$\begin{bmatrix} Q + D_k^2 & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ -\Delta y_k \end{bmatrix} = \begin{bmatrix} r_k^1 + X_k^{-1} r_k^3 \\ r_k^2 \end{bmatrix}, \tag{6}$$

where $D_k = X_k^{-\frac{1}{2}} Z_k^{\frac{1}{2}}$.

The linear system (6) is known as the "augmented system" and is typically solved by performing a Bunch-Parlett or Bunch-Kaufman factorization [10,11]. When the direct factorization is not possible because of memory limitations, iterative techniques for such systems are used [2,17]. An alternative to solving the augmented system is the so-called "normal equations" approach which reduces the augmented system to a symmetric positive definite system of the form $A(Q + D_k^2)^{-1}A^T\Delta y_k = r$ which can be solved by using Cholesky-based factorizations or iterative methods [2,12,29]. The normal equations approach is very popular in the context of linear programming $(Q = 0)$, but rarely used for quadratic programming because the factorization of $Q+D_k^2$ and the multiple solves with its factors tend to be computationally burdensome.

2.1 The Schur complement decomposition

We now show how the parallelization of linear algebra can be achieved for two-stage SP problems via the Schur complement mechanism. The technique parallelizes the solution of the augmented system. For similar approaches for the solutions of normal equations for linear programming we refer the reader to [8,6,45].

As we mentioned in the introduction, the two-stage stochastic programming problem (1)-(2) is equivalent, in the discrete case, or approximated, in the continuous case, by a deterministic optimization problem with a particular form given by the staircase structure of the constraints' Jacobian and separability of the objective function. Any such deterministic problem can be formulated as

$$
\begin{array}{lllll}
\min & \multicolumn{4}{l}{\left(\frac{1}{2}x_0^TQ_0x_0+c_0^Tx_0\right)+\sum_{i=1}^{N}p_i\left(\frac{1}{2}x_i^TQ_ix_i+c_i^Tx_i\right)} \\
\text{subj to} & T_0x_0 & & & =b_0, \\
 & T_1x_0+ & W_1x_1 & & =b_1, \\
 & T_2x_0+ & \qquad W_2x_2 & & =b_2, \\
 & \vdots & \qquad\qquad \ddots & & \vdots \\
 & T_Nx_0+ & & W_Nx_N & =b_N, \\
 & \multicolumn{4}{l}{x_0\ge 0,\ x_1\ge 0\ ,x_2\ge 0,\dots x_N\ge 0.}
\end{array}
\tag{7}
$$

Here we use x_0 for the first-stage variables and $x_1,\dots,x_N$ for the variables of the second-stage subproblems corresponding to a sample $\xi_1,\dots,\xi_N$ withdrawn from $\xi(\omega)$. The same notation convention is used for the problem's data: Q, c, b, and T from the first-stage problem (1) are renamed to Q_0, c_0, b_0, and T_0, and Q_i, c_i, b_i, T_i, and W_i, $i=1,\dots,N$, correspond to second-stage data $Q(\omega)$, $c(\omega)$, $b(\omega)$, $T(\omega)$, and $W(\omega)$.

The system (4) corresponding to (7) has the following form:

$$
\begin{aligned}
Q_0x_0-T_0^Ty_0-\tfrac{1}{N}\sum_{i=1}^{N}T_i^Ty_i-z_0 &= -c_0, \\
\tfrac{1}{N}Q_ix_i-\tfrac{1}{N}W_i^Ty_i-\tfrac{1}{N}z_i &= -\tfrac{1}{N}c_i, \\
T_0x_0 &= b_0, \\
T_ix_0+W_ix_i &= b_i, \qquad i=1,2,\dots,N, \\
x_0z_0 &= \mu e, \\
x_iz_i &= \tfrac{1}{N}\mu e, \quad i=1,\dots,N, \\
x_i,z_i &\ge 0, \qquad i=0,1,\dots,N.
\end{aligned}
\tag{8}
$$

In order to be consistent with the optimality conditions for stochastic programming problems presented in [51], the multipliers $y_1,y_2,\dots,y_N$ and $z_1,z_2,\dots,z_N$ were scaled by N in the derivation of (8).

The linear system (6) solved at each iteration of the interior-point algorithm becomes in the case of the two-stage SAA SP problem:

$$
\begin{bmatrix}
Q_0+D_0^2 & & & & T_0^T & \frac{1}{N}T_1^T & \dots & \frac{1}{N}T_N^T \\
 & \frac{1}{N}(Q_1+D_1^2) & & & 0 & \frac{1}{N}W_1^T & \dots & 0 \\
 & & \ddots & & \vdots & \vdots & \ddots & \vdots \\
 & & & \frac{1}{N}(Q_N+D_N^2) & 0 & 0 & \dots & \frac{1}{N}W_N^T \\
T_0 & 0 & \dots & 0 & & & & \\
\frac{1}{N}T_1 & \frac{1}{N}W_1 & \dots & 0 & & & & \\
\vdots & \vdots & \ddots & \vdots & & & & \\
\frac{1}{N}T_N & 0 & \dots & \frac{1}{N}W_N & & & &
\end{bmatrix}
\begin{bmatrix}
\Delta x_0 \\ \Delta x_1 \\ \vdots \\ \Delta x_N \\ -\Delta y_0 \\ -\Delta y_1 \\ -\Delta y_2 \\ \vdots \\ -\Delta y_N
\end{bmatrix}
= r,
\tag{9}
$$

where $D_i=X_i^{-\frac{1}{2}}Z_i^{\frac{1}{2}}$ and r is computed accordingly. From now on we will use the notation $\bar{Q}_i=Q_i+D_i^2$, $i=0,1,2,\dots,N$.

A permutation of the blocks of the above linear system is needed to obtain the linear system (10), which is specific to stochastic programming and suitable for parallelization. Let us first consider the following permutation mapping

$$
\pi=\begin{pmatrix}
1 & 2\ 3\dots & N & N+1 & N+2 & N+3\dots & 2N+1 & 2N+2 \\
2N+1 & 1\ 3\dots & 2N-3 & 2N-1 & 2N+2 & 2\quad\dots & 2N-2 & 2N
\end{pmatrix}.
$$

In this 2-line notation, the first row in π contains the arguments 1 to $2N+2$, and the second row contains images $\pi(1),\dots,\pi(2N+2)$ of the permutation mapping. Also consider the matrix Π of the same size and block structure as the matrix from (9) whose blocks are 0 except $(i,\pi(i))$ blocks, $i=1,\dots,2N+2$, which are identity matrices.

If the permutation Π is applied to both the equations (rows) and the unknowns (columns) of (9), then the equivalent linear system (10) is obtained.

$$\begin{bmatrix} \frac{1}{N}\bar{Q}_1 & \frac{1}{N}W_1^T & & & & 0 & 0 \\ \frac{1}{N}W_1 & 0 & & & & \frac{1}{N}T_1 & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & \frac{1}{N}\bar{Q}_N & \frac{1}{N}W_N^T & 0 & 0 \\ & & & \frac{1}{N}W_N & 0 & \frac{1}{N}T_N & 0 \\ 0 & \frac{1}{N}T_1^T & \dots & 0 & \frac{1}{N}T_N^T & \bar{Q}_0 & T_0^T \\ 0 & 0 & \dots & 0 & 0 & T_0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ -\Delta y_1 \\ \vdots \\ \Delta x_N \\ -\Delta y_N \\ \Delta x_0 \\ -\Delta y_0 \end{bmatrix} = \Pi r. \tag{10}$$

When the Schur complement of the lower right 2-by-2 block in the system's matrix is used, a scenario-based decomposition is obtained. The same mechanism is used in [25,26,28,55] to achieve the parallelization of the problem. We briefly show how the factorization as well as the forward and back substitutions phases can be parallelized; we refer the reader to the abovementioned papers for more details.

We further simplify the notation and denote by $K_1, K_2, \dots, K_N, K_0$ the diagonal blocks and by $B_1, B_2, \dots, B_N$ the bordering blocks of the system matrix of (10). Hence

$$K_i = \begin{bmatrix} \frac{1}{N}\bar{Q}_i & \frac{1}{N}W_i^T \\ \frac{1}{N}W_i & 0 \end{bmatrix}, \qquad K_0 = \begin{bmatrix} \bar{Q}_0 & T_0^T \\ T_0 & 0 \end{bmatrix},$$
$$B_i = \begin{bmatrix} 0 & 0 \\ \frac{1}{N}T_i & 0 \end{bmatrix}, \qquad i = 1,2,\dots,N.$$

Let $\Delta u_i = \begin{bmatrix} \Delta x_i^T & -\Delta y_i^T \end{bmatrix}^T$, $i=0,1,\dots,N$, $\Delta u = \begin{bmatrix} \Delta u_1^T & \dots & \Delta u_N^T & \Delta u_0^T \end{bmatrix}^T$, and let Πr be of the form $\begin{bmatrix} r_1^T & \dots & r_N^T & r_0^T \end{bmatrix}^T$.

With the new notation, the matrix of the linear system (10) becomes

$$K = \begin{bmatrix} K_1 & & & B_1 \\ & \ddots & & \vdots \\ & & K_N & B_N \\ B_1^T & \dots & B_N^T & K_0 \end{bmatrix}. \tag{11}$$

Since K is symmetric, it can be factorized as LDL^T, where L is a unit lower triangular matrix and D is block diagonal matrix with 1×1 and 2×2 diagonal blocks [10,11].

One can easily verify that L and D have the following particular structures,

$$L = \begin{bmatrix} L_1 & & & \\ \vdots & \ddots & & \\ & & L_N & \\ L_{N1} & \dots & L_{NN} & L_c \end{bmatrix}, \quad D = \begin{bmatrix} D_1 & & & \\ & \ddots & & \\ & & D_N & \\ & & & D_c \end{bmatrix},$$

where

$$L_i D_i L_i^T = K_i,\ i = 1, \ldots, N, \tag{12}$$

$$L_{Ni} = B_i^T L_i^{-T} D_i^{-1},\ i = 1 \ldots, N, \tag{13}$$

$$L_c D_c L_c^T = C = K_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} B_i. \tag{14}$$

Observe that C defined above is the Schur complement of the first-stage Hessian block K_0 in the entire Hessian matrix K.

Consequently, the following steps are needed to solve $K\Delta u = r$:

$$w_i = L_i^{-1} r_i,\ i = 1, \ldots, N, \tag{15}$$

$$w_0 = L_c^{-1}\left(r_0 - \sum_{i=1}^{N} L_{Ni} w_i\right), \tag{16}$$

$$v_0 = D_c^{-1} w_0,\ v_i = D_i^{-1} w_i,\ i = 1, \ldots, N, \tag{17}$$

$$\Delta u_0 = L_c^{-T} v_0, \tag{18}$$

$$\Delta u_i = L_i^{-T}(v_i - L_{Ni}\Delta u_0),\ i = 1, \ldots, N. \tag{19}$$

These operations can be divided into four phases:

– *Factorization*: (12), (13) and (14);
– *Forward substitution*: steps (15) and (16);
– *Diagonal solve*: step (17);
– *Back substitution*: steps (18) and (19).

3 The preconditioning

In this section we first introduce and analyze the spectral properties of a preconditioner for the symmetric positive definite $(1, 1)$ block of the Schur complement matrix C. The preconditioner approximates the $(1, 1)$ block of C by using a subset of the existing scenarios; it is called "stochastic preconditioner". We then present two approaches for iteratively solving $C\Delta u_0 = r_0$. The first approach algebraically incorporates the stochastic preconditioner into a symmetric indefinite preconditioner for C. The resulting preconditioner is known in the literature as the constraint preconditioner. The second approach uses the stochastic preconditioner with a projected conjugate gradient method to solve a reduced linear system.

The expression (14) of the Schur complement matrix C can be rewritten as

$$C = \begin{bmatrix} \bar{Q}_0 & T_0^T \\ T_0 & 0 \end{bmatrix} - \frac{1}{N}\sum_{i=1}^{N} \begin{bmatrix} 0 & T_i^T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{Q}_i & W_i^T \\ W_i & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ T_i & 0 \end{bmatrix}. \tag{20}$$

We use the following well-known matrix inversion identity to express the Schur complement matrix C in terms of the data of the original problem.

Lemma 1 *For any nonsingular matrix A and any full-rank matrix B the following matrix identity holds:*

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} - A^{-1}B^T\left(BA^{-1}B^T\right)^{-1}BA^{-1} & A^{-1}B^T\left(BA^{-1}B^T\right)^{-1} \\ \left(BA^{-1}B^T\right)^{-1}BA^{-1} & -\left(BA^{-1}B^T\right)^{-1} \end{bmatrix}.$$

Proof By direct verification.

By applying the above identity to each of the inner blocks of the summation from (20), C can be written as

$$C = \begin{bmatrix} \bar{Q}_0 + \frac{1}{N}\sum_{i=1}^{N} T_i^T\left(W_i\bar{Q}_i^{-1}W_i^T\right)^{-1}T_i & T_0^T \\ T_0 & 0 \end{bmatrix}. \tag{21}$$

3.1 The stochastic preconditioner

Let us denote the $(1,1)$ block from (21) by S_N and observe that

$$S_N = \left(Q_0 + D_0^2\right) + \frac{1}{N}\sum_{i=1}^{N} M_i,$$

where

$$M_i = T_i^T\left(W_i\left(Q_i + D_i^2\right)^{-1}W_i^T\right)^{-1}T_i.$$

Let $\xi_{k_1}, \xi_{k_2}, \ldots, \xi_{k_n}$ be an independent identically distributed (IID) sample withdrawn from $\xi_1, \xi_2, \ldots, \xi_N$ with $n << N$, and denote

$$S_n = \left(Q_0 + D_0^2\right) + \frac{1}{n}\sum_{i=1}^{n} M_{k_i}.$$

We propose S_n as a preconditioner for S_N.

In the remainder of this section we show that the probability that the eigenvalues of the preconditioned matrix $S_n^{-1}S_N$ are outside an ϵ-ball centered at 1 exponentially approaches 0 as n increases. In other words, the eigenvalues of the preconditioned matrix cluster around 1 "exponentially" with the number of subsamples. The fact that the eigenvalues cluster around 1 is obvious since for $n = N$ the preconditioned matrix is exactly the identity matrix. The exponential rate of clustering will make the preconditioner effective since n is taken to be much smaller than N for the sake of parallel efficiency.

In particular we regard the entries of matrices M_i, $i = 1, \ldots, N$ as independent identically distributed realizations of the corresponding entries of the random matrix

$$M(\omega) = T^T(\omega)\left(W(\omega)\left(Q(\omega) + D^2(\omega)\right)^{-1}W^T(\omega)\right)^{-1}T(\omega), \tag{22}$$

and use the theory of large deviations to obtain the exponential convergence bounds. One can see that $M(\omega)$ depends on the barrier parameter $\mu > 0$ through the diagonal matrix $D(\omega)$. The analysis of this section is pointwise with μ; that is, the convergence results hold for any $\mu > 0$, but the exponential bounds change with μ.

We now present the assumptions on the random data $\xi(\omega)$ that are needed to ensure that the entries of $M(\omega)$ are bounded random variables. We first assume the boundedness of the random data of the two-stage SP problem:

$(A1)$ There exists $U > 0$ such that $\|\xi(\omega)\| < U$, for any $\omega \in \Omega$.

Second, we require the singular values of the recourse matrix $W(\omega)$ to be bounded away from 0 and the singular values of the technology matrix $T(\omega)$ to stay bounded uniformly with ω:

$(A2)$ There exists $\sigma_m^W > 0$ such that

$$0 < \sigma_m^W \leq \sigma_{min}(W(\omega)), \text{ for any } \omega \in \Omega.$$

$(A3)$ There exists $\sigma_M^T \in \mathbb{R}$ such that

$$\sigma_{max}(T(\omega)) \leq \sigma_M^T \text{ for any } \omega \in \Omega.$$

Another, obvious in some way, assumption is that the quadratic objective matrix $Q(\omega)$ should also stay bounded uniformly with ω, namely

$(A4)$ There exists $\lambda_M^Q > 0$ such that

$$\lambda_{max}(Q(\omega)) \leq \lambda_M^Q \text{ for any } \omega \in \Omega.$$

Note that in the case of a discrete or compact probability space Ω, Assumptions $(A2)$, $(A3)$, and $(A4)$ follow from Assumption $(A1)$. Therefore, the conditions under which we analyze the preconditioner hold for a large class of applications.

Furthermore, the homeomorphism of the mapping defining the central path (see [31] or [43]) and Assumption (A4) allow us to consider the second-stage components of the central path, namely $x_1(\mu), \ldots, x_N(\mu)$ and $z_1(\mu), \ldots, z_N(\mu)$ as IID realizations of some random vectors $\chi = \chi_\mu(\omega)$ and $\zeta = \zeta_\mu(\omega)$ for any $\mu > 0$. Consequently, we can regard the diagonal matrices D_i, $i = 1, \ldots, N$, as realizations of the random matrix $D_\mu(\omega) = \chi_\mu^{-\frac{1}{2}} \zeta_\mu^{\frac{1}{2}}$. The boundedness of $D_\mu(\omega)$ follows from the fact that $x_1, \ldots, x_N$ must be positive since they satisfy (8) with $\mu > 0$.

As we mentioned in Section 2, path-following algorithms follow the central path approximately, and the iterates are situated in a neighborhood of the central path. The proof of the randomness and boundedness of $D_\mu(\omega)$ is intractable in the case of Mehrotra's algorithm because this method does not use an explicit neighborhood of the central path. However, the algorithm maintains the positiveness of the updates by backing off by a fixed factor whenever the boundary of the positive orthant is reached. Therefore, we assume that the entries of $D(\omega)$ are bounded random variables:

$(A5)$ For a given $\mu > 0$, there exists positive numbers d_m^μ and d_M^μ such that

$$0 < d_m^\mu \leq \left[D^2(\omega)\right]_{ii} \leq d_M^\mu, \text{ for any } i \text{ and } \omega \in \Omega.$$

We now present the spectral results that are used in the proof of Lemma 3. By $\|\cdot\|$ we denote the 2-norm of a vector or matrix.

Lemma 2 *Let $Q \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix and $A \in \mathbb{R}^{m \times n}$ be a full-rank matrix. The following statements hold:*

(i) The extreme eigenvalues of Q are characterized by

$$\lambda_{min}(Q) = \underset{x\neq 0}{min} \frac{x^T Q x}{\|x\|^2} \text{ and } \lambda_{max}(Q) = \underset{x\neq 0}{max} \frac{x^T Q x}{\|x\|^2} = \|Q\|.$$

(ii) The largest singular value of A is characterized by

$$\|A\| = \underset{x\neq 0}{max} \frac{\|Ax\|}{\|x\|} = \sigma_{max}(A) = \underset{y\neq 0}{max} \frac{\|A^T y\|}{\|y\|} = \|A^T\|.$$

When $m \leq n$, the smallest singular value of A satisfies

$$\sigma_{min}(A) = \underset{x\neq 0}{min} \frac{\|A^T x\|}{\|x\|}.$$

(iii) In the case $m \leq n$, the extreme eigenvalues of AQA^T satisfy

$$\lambda_{min}(Q)\, \sigma^2_{min}(A) \leq \lambda_{min}\left(AQA^T\right) \leq \lambda_{max}\left(AQA^T\right) \leq \lambda_{max}(Q)\, \sigma^2_{max}(A).$$

Proof (i) A simple proof can be obtained based on the symmetric Schur decomposition of Q (page 393 in [22]) or based on the Courant-Fischer minimax theorem (also in [22], page 394).

(ii) See [22], page 71.

(iii) For any $y \in \mathbb{R}^m$, from (i) and (ii) we can write

$$y^T AQA^T y \geq \lambda_{min}(Q)\|A^T y\|^2 \geq \lambda_{min}(Q)\sigma^2_{min}(A)\|y\|^2 \text{ and}$$
$$y^T AQA^T y \leq \lambda_{max}(Q)\|A^T y\|^2 \leq \lambda_{max}(Q)\sigma^2_{max}(A)\|y\|^2.$$

These inequalities and (i) applied to AQA^T imply that $\lambda_{min}(Q)\sigma^2_{min}(A)$ is a lower bound for $\lambda_{min}(AQA^T)$ and $\lambda_{max}(Q)\sigma^2_{max}(A)$ is an upper bound for $\lambda_{max}(Q)\sigma^2_{max}(A)$, and hence inequalities from (iii) hold.

Lemma 3 *Under Assumptions (A1)-(A5) the matrix*

$$M(\omega) = T^T(\omega)\left(W(\omega)\left(Q(\omega) + D^2(\omega)\right)^{-1} W^T(\omega)\right)^{-1} T(\omega)$$

is well defined, and each of its entries is a bounded random variable.

Proof Since the entries of $D^2(\omega)$ are positive and the $Q(\omega)$ is positive semidefinite, we have that (see [22, Theorem 8.1.5])

$$d^\mu_m \leq \lambda_{min}\left(Q(\omega) + D^2(\omega)\right) \leq \lambda_{max}\left(Q(\omega) + D^2(\omega)\right) \leq d^\mu_M + \lambda^Q_M, \forall\omega, \tag{23}$$

which implies that the matrix $Q(\omega) + D^2(\omega)$ is invertible. From $(A2)$ we obtain that $W(\omega)\left(Q(\omega) + D^2(\omega)\right)^{-1} W^T(\omega)$ is also invertible. This shows that $M(\omega)$ is well defined and its components are random variables.

We now prove the componentwise boundedness. Observe that (23) also allows us to deduce that

$$\lambda_{min}\left(\left(Q(\omega) + D^2(\omega)\right)^{-1}\right) \geq \frac{1}{d^\mu_M + \lambda^Q_M}, \forall\omega. \tag{24}$$

Then by (iii) of Lemma 2 the following bound exists for the smallest eigenvalue of the matrix $W(\omega)\left(Q(\omega)+D^2(\omega)\right)^{-1}W^T(\omega)$,

$$\lambda_{min}\left(W(\omega)\left(Q(\omega)+D^2(\omega)\right)^{-1}W^T(\omega)\right)\geq\frac{\left(\sigma_m^W\right)^2}{d_M^\mu+\lambda_M^Q},\forall\omega.$$

Therefore,

$$\lambda_{max}\left(\left(W(\omega)\left(Q(\omega)+D^2(\omega)\right)^{-1}W^T(\omega)\right)^{-1}\right)\leq\frac{d_M^\mu+\lambda_M^Q}{\left(\sigma_m^W\right)^2},\forall\omega. \qquad (25)$$

By the submultiplicative property of the 2-norms and based on (i) and (ii) of Lemma 2 and the bound (25), the 2-norm of $M(\omega)$ can be bounded as follows

$$\begin{aligned}\|M(\omega)\| &\leq \left\|T^T(\omega)\right\|\left\|\left(W(\omega)\left(Q(\omega)+D^2(\omega)\right)^{-1}W^T(\omega)\right)\right\|\|T(\omega)\|\\ &\leq\frac{\left(\sigma_M^T\right)^2}{\left(\sigma_m^W\right)^2}\left(d_M^\mu+\lambda_M^Q\right),\forall\omega.\end{aligned}$$

The boundedness of the entries of $M(\omega)$ follows immediately by writing

$$\left|[M(\omega)]_{ij}\right|=\left|e_i^TM(\omega)e_j\right|\leq\|M(\omega)\|\leq\frac{\left(\sigma_M^T\right)^2}{\left(\sigma_m^W\right)^2}\left(d_M^\mu+\lambda_M^Q\right),\forall\omega,$$

and hence the lemma is proved.

Let us denote the expected value, or the population mean, of $\mathcal{S}(\omega)=(Q_0+D_0^2)+M(\omega)$ by $S\,(=\mathbb{E}[\mathcal{S}(\omega)])$ and observe that the matrix S_N is the sample mean. This remark allows us to use the Hoeffding's inequality [51] from the theory of large deviations as a base for the proof that the eigenvalues of $S_N^{-1}S$ cluster exponentially fast with N around 1. The exponential clustering will also hold for the preconditioned matrix $S_n^{-1}S_N$ since, under the IID sampling scheme used to obtain $\xi_{k_1},\ldots,\xi_{k_n}$, the matrix S_n is a sample mean of S_N. In this case S_N is the expected value taken with respect to the empirical distribution given by $\xi_1,\ldots,\xi_N$.

Lemma 3 also implies the boundedness of $\mathcal{S}(\omega)$, and we simplify the notation by letting $L>0$ (depending on λ_M^Q, σ_m^W, σ_M^T, and d_M^μ as per Lemma 3) such that

$$\left|[\mathcal{S}(\omega)]_{ij}\right|<L,\text{ for any }\omega\in\Omega.$$

Hoeffding's inequality (see Chapter 7 of [51]) is a concentration result showing that the probability that the sample mean is close to the population mean approaches 1 exponentially as the number of samples increases. In this case, it states that for any $\epsilon>0$,

$$Pr\left(\left|[S_N]_{ij}-S_{ij}\right|\geq\epsilon\right)\leq2\exp\left(-\frac{\epsilon^2N}{2L^2}\right). \qquad (26)$$

The following lemma characterizes the distance between the components of the identity matrix and the preconditioned matrix $S_N^{-1}S$. We denote by S and S_N and by $\|S^{-1}\|_{max}$ the largest magnitude of the components of S^{-1}. Note that the invertibility of S follows from Lemma 3. Also we recall that the size of S and S_N is denoted by n_{x_0}.

Lemma 4 *For any $\epsilon > 0$,*

$$Pr\left(\left|\left[I - S^{-1}S_N\right]_{ij}\right| \geq \epsilon\right) \leq 2n_{x_0}\exp\left(-\frac{\epsilon^2 N}{2n_{x_0}^2 L^2 \|S^{-1}\|_{max}^2}\right). \tag{27}$$

Proof Since $I - S^{-1}S_N = S^{-1}(S - S_N)$, we have

$$\left|\left[I - S^{-1}S_N\right]_{ij}\right| = \left|\sum_{k=1}^{n_{x_0}} \left[S^{-1}\right]_{ik} [S - S_N]_{kj}\right| \leq \|S^{-1}\|_{max} \sum_{k=1}^{n_{x_0}} \left|[S - S_N]_{kj}\right|,$$

which allows us to write

$$Pr\left(\left|\left[I - S^{-1}S_N\right]_{ij}\right| \geq \epsilon\right) \leq Pr\left(\sum_{k=1}^{n_{x_0}} \left|[S - S_N]_{kj}\right| \geq \frac{\epsilon}{\|S^{-1}\|_{max}}\right). \tag{28}$$

In bounding the right term in the above inequality we use the following inequality holding for any random variables $Y_1, Y_2, \ldots, Y_p$, $p \geq 1$:

$$Pr\left(\sum_{k=1}^{p} Y_i \geq a\right) \leq \sum_{k=1}^{p} Pr\left(Y_i \geq \frac{a}{p}\right). \tag{29}$$

For a proof see [51, Inequality (7.101)]. Then (28) can be further bounded as follows:

$$\begin{aligned} Pr\left(\left|\left[I - S^{-1}S_N\right]_{ij}\right| \geq \epsilon\right) &\leq \sum_{k=1}^{n_{x_0}} Pr\left(\left|[S - S_N]_{kj}\right| \geq \frac{\epsilon}{n_{x_0}\|S^{-1}\|_{max}}\right) \\ &\leq \sum_{k=1}^{n_{x_0}} 2\ \exp\left(-\frac{\epsilon^2 N}{2n_{x_0}^2 L^2 \|S^{-1}\|_{max}^2}\right) \text{(by (26))} \\ &= 2n_{x_0}\exp\left(-\frac{\epsilon^2 N}{2n_{x_0}^2 L^2 \|S^{-1}\|_{max}^2}\right), \end{aligned}$$

which proves the thesis.

Lemma 5 *Let $\epsilon > 0$. If the symmetric matrices $A, B \in \mathbb{R}^{p \times p}$ satisfy*

$$Pr\left(\left|A_{ij} - B_{ij}\right| \geq \epsilon\right) \leq c\ \exp\left(-C\epsilon^2\right), \forall i, j \in \{1, 2, \ldots, p\},$$

for some constants $c > 0$ and $C > 0$ that does not depend on ϵ, then the eigenvalues of A and B are characterized by

$$Pr\left(|\lambda_k(A) - \lambda_k(B)| \geq \epsilon\right) \leq cp^2\ \exp\left(\frac{-C\epsilon^2}{p^2}\right).$$

Proof By the Wielandt-Hoffman theorem (page 395 in [22]) we have that

$$\sum_k \left(\lambda_k(A) - \lambda_k(B)\right)^2 \leq \|A - B\|_F^2. \tag{30}$$

Furthermore, we can write

$$\begin{aligned}
Pr\left(|\lambda_k(A)-\lambda_k(B)|\geq\epsilon\right) &= Pr\left(|\lambda_k(A)-\lambda_k(B)|^2\geq\epsilon^2\right)\\
&\leq Pr\left(\sum_{k=1}^{p}(\lambda_k(A)-\lambda_k(B))^2\geq\epsilon^2\right)\\
&\leq Pr\left(\|A-B\|_F^2\geq\epsilon^2\right)\quad\text{(by (30))}\\
&= Pr\left(\sum_{i,j=1}^{p}|A_{ij}-B_{ij}|^2\geq\epsilon^2\right)\\
&\leq \sum_{i,j=1}^{p} Pr\left(|A_{ij}-B_{ij}|^2\geq\epsilon^2/p^2\right)\quad\text{(by (29))}\\
&= \sum_{i,j=1}^{p} Pr\left(|A_{ij}-B_{ij}|\geq\epsilon/p\right)\leq cp^2\exp\left(\frac{-C\epsilon^2}{p^2}\right),
\end{aligned}$$

which completes the proof.

The following result shows that the eigenvalues of the preconditioned matrix $S_N^{-1}S$ cluster around 1 exponentially with the sample size N.

Theorem 1 *For any $\epsilon>0$ the eigenvalues of $S_N^{-1}S$ satisfy*

$$Pr\left(|\lambda(S_N^{-1}S)-1|\geq\epsilon\right)\leq 2n_{x_0}^3\exp\left(-\frac{N\left(\frac{\epsilon}{1+\epsilon}\right)^2}{2n_{x_0}^4L^2\|S^{-1}\|_{max}^2}\right).$$

Proof Let us first consider an arbitrary $\epsilon\in(0,1)$. By applying Lemma 5 to the matrices I and $S^{-1}S_N$ and using the inequality (27) given in Lemma 4, we can write

$$Pr\left(|\lambda(S^{-1}S_N)-1|\geq\epsilon\right)\leq 2n_{x_0}^3\exp\left(-\frac{\epsilon^2N}{2n_{x_0}^4L^2\|S^{-1}\|_{max}^2}\right).$$

Since $Pr\left(|\lambda(S^{-1}S_N)-1|\geq\epsilon\right)=1-Pr\left(|\lambda(S^{-1}S_N)-1|<\epsilon\right)$ and $\lambda(S^{-1}S_N)=1/\lambda(S_N^{-1}S)$, the above inequality can be transformed to

$$1-Pr\left(\frac{1}{1+\epsilon}<\lambda(S_N^{-1}S)<\frac{1}{1-\epsilon}\right)\leq 2n_{x_0}^3\exp\left(-\frac{\epsilon^2N}{2n_{x_0}^4L^2\|S^{-1}\|_{max}^2}\right),$$

or,

$$1-Pr\left(1-\frac{\epsilon}{1+\epsilon}<\lambda(S_N^{-1}S)<1+\frac{\epsilon}{1-\epsilon}\right)\leq 2n_{x_0}^3\exp\left(-\frac{\epsilon^2N}{2n_{x_0}^4L^2\|S^{-1}\|_{max}^2}\right).\quad(31)$$

It can be easily verified that $1-\frac{\epsilon}{1-\epsilon}<1-\frac{\epsilon}{1+\epsilon}$ for any $\epsilon\in(0,1)$. Therefore,

$$Pr\left(1-\frac{\epsilon}{1-\epsilon}<\lambda(S_N^{-1}S)<1+\frac{\epsilon}{1-\epsilon}\right)\geq Pr\left(1-\frac{\epsilon}{1+\epsilon}<\lambda(S_N^{-1}S)<1+\frac{\epsilon}{1-\epsilon}\right),$$

and, based on (31), we obtain that

$$1-Pr\left(1-\frac{\epsilon}{1-\epsilon}<\lambda(S_N^{-1}S)<1+\frac{\epsilon}{1-\epsilon}\right)\leq 2n_{x_0}^3\exp\left(-\frac{\epsilon^2N}{2n_{x_0}^4L^2\|S^{-1}\|_{max}^2}\right),$$

or, equivalently,

$$Pr\left(|\lambda(S_N^{-1}S)-1|\geq\frac{\epsilon}{1-\epsilon}\right)\leq 2n_{x_0}^3\exp\left(-\frac{\epsilon^2N}{2n_{x_0}^4L^2\|S^{-1}\|_{max}^2}\right).\quad(32)$$

Consider now an arbitrary $\epsilon > 0$. We apply (32) to $\epsilon/(1+\epsilon) \in (0,1)$, to obtain

$$Pr(\left(|\lambda(S_N^{-1}S) - 1| \geq \frac{\epsilon/(1+\epsilon)}{1-\epsilon/(1+\epsilon)} = \epsilon\right) \leq 2n_{x_0}^3 \exp\left(-\frac{N\left(\frac{\epsilon}{1+\epsilon}\right)^2}{2n_{x_0}^4 L^2 \|S^{-1}\|_{max}^2}\right),$$

which completes the proof

As we mentioned before, the bound of Theorem 1 holds for the preconditioner of interest, $S_n^{-1}S_N$, since S_n is the sample mean of the population mean S_N in the empirical distribution given by $\xi_1, \ldots, \xi_N$. More specifically, we have

$$Pr\left(|\lambda(S_n^{-1}S_N) - 1| \geq \epsilon\right) \leq 2n_{x_0}^3 \exp\left(-\frac{n\left(\frac{\epsilon}{1+\epsilon}\right)^2}{2n_{x_0}^4 L^2 \|S_N^{-1}\|_{max}^2}\right). \tag{33}$$

A similar observation shows that the exponential clustering of the eigenvalues of $S_n^{-1}S_N$ also occurs for stochastic programming problems with non-equiprobable scenarios. More specifically, let us assume that there are N scenarios $\xi_1, \ldots, \xi_N$, with corresponding probabilities p_i, $i = 1, \ldots, N$, $p_1 + \ldots + p_N = 1$. Observe that we now have

$$S_N = \left(Q_0 + D_0^2\right) + \sum_{i=1}^{N} p_i\, M_i.$$

The key point in this case is to compute the preconditioner based on a set of n IID samples $\xi_{k_1}, \ldots, \xi_{k_n}$ withdrawn from the empirical distribution of $\xi_1, \ldots, \xi_N$ (with corresponding probabilities $p_1, \ldots, p_N$). The sampling can be easily done by using Monte Carlo sampling [51], but any sampling that yields an IID subsample can be used. The preconditioner is computed based on the same formula as in the equiprobable case, that is

$$S_n = \left(Q_0 + D_0^2\right) + \frac{1}{n}\sum_{i=1}^{n} M_{k_i}.$$

Under this sampling scheme, S_n is still a sample mean of the population mean S_N and, therefore, (33) holds.

3.2 The constraint preconditioner

A popular preconditioning technique for saddle-point matrices such as C is the class of constraint preconditioners [33], that incorporates an existing preconditioner of the $(1,1)$ block together with the $(1,2)$ and $(2,1)$ blocks of the original saddle-point system.

A constraint preconditioner corresponding to C is

$$M = \begin{bmatrix} S_n & T_0^T \\ T_0 & 0 \end{bmatrix}. \tag{34}$$

The spectral analysis from [3] (see proof of Theorem 4.1 of the paper) reveals that the preconditioned matrix $M^{-1}C$ has an eigenvalue at 1 with order of multiplicity $2m_{y_0}$ and $n_{x_0} - m_{y_0}$ real eigenvalues satisfying

$$0 < \lambda_{min}(S_n^{-1}S_N) \leq \lambda(M^{-1}C) \leq \lambda_{max}(S_n^{-1}S_N),$$

where m_{y_0} are the number of equality constraints from the first-stage problem (number of rows of T_0). Hence the constraint preconditioner (34) possesses the same exponential clustering of the eigenvalues around 1 as the stochastic preconditioner we introduced in the previous section.

Algorithm *DSC* for Process $p \in \mathcal{P}$

```
Factorization phase
1.    For each $i \in \mathcal{N}_p$ factorize $L_i D_i L_i^T = K_i$;
2.    Compute $C_p = -\sum_{i \in \mathcal{N}_p} B_i^T K_i^{-1} B_i$, all-reduce $C = \sum_{r \in \mathcal{P}} C_r$,
      and compute $C = C + K_0$;
3*.   Factorize the Schur complement $L_c D_c L_c^T = C$;
Back substitution phase
4.    For each $i \in \mathcal{N}_p$ solve $w_i = L_i^{-1} r_i$;
5.    Compute $\sum_{i \in \mathcal{N}_p} L_{Ni} w_i$ and all-reduce $\sum_{i=1}^{N} L_{Ni} w_i$;
6*.   Solve $w_0 = L_c^{-1}(r_0 - \sum_{i=1}^{N} L_{Ni} w_i)$;
Diagonal solve phase
7.    For each $i \in \mathcal{N}_p$ solve $v_i = D_i^{-1} w_i$;
8*.   Solve $v_0 = D_c^{-1} w_0$;
Forward substitution phase
9*.   Solve $\Delta u_0 = L_c^{-T} v_0$;
10.   For each $i \in \mathcal{N}_p$ solve $\Delta u_i = L_i^{-T}(v_i - L_{Ni} \Delta u_0)$.
```

4 Parallel implementation

The linear solve $K\Delta u = \Pi r$ from Section 2 is suitable to a scenario-based parallelization. While the factorizations and triangular solves involving the second-stage variable are fully parallelizable, the factorization and triangular solves involving the first stage variable are not because the Schur complement C given by (14) requires data from all scenarios.

Our parallel implementation distributes both the data and the work corresponding to second-stage variables across multiple processes and uses the Message Passing Interface (MPI) as the underlying mechanism for communication between processes. The Schur complement C is stored and factorized on all processes. This approach causes a bottleneck that is addressed by the preconditioning technique presented in Section 3.

Scenarios are evenly assigned to available computational units by solving a number partitioning problem. More exactly, if $\mathcal{P} = \{1, 2, \ldots, P\}$ denotes the set of available computational units, and if there are N scenarios, each with positive load l_i, $i = \{1, 2, \ldots, N\}$, the number partitioning problem consists of finding a partition $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_P$ of $\{1, 2, \ldots, N\}$ such that the differences between

$$\sum_{i \in \mathcal{N}_1} l_i, \sum_{i \in \mathcal{N}_2} l_i, \ldots, \sum_{i \in \mathcal{N}_P} l_i$$

are as small as possible, ideally zero. The load l_i is the wall-clock time needed to perform work associated with scenario i at the previous interior-point iteration. For the first interior-point iteration the load is computed based on the size and the fill-in of the scenario's data.

We first list the algorithm that uses the direct factorization of the Schur complement matrix to solve the linear system $K\Delta u = r$ given in Section 2. We call this algorithm *DSC* (direct Schur complement); it solves a two-stage problem involving N scenarios by using P processes. Once the partition $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_P$ of $\{1, 2, \ldots, N\}$ is obtained, any process $p \in \mathcal{P}$ performs the set of operations listed in Algorithm *DSC*. The first-stage operations that are performed on all processes are marked with a "*".

4.1 Parallel implementation of the preconditioning

The factorization from step 3^* and the solves from steps 6^*, 8^*, and 9^* of Algorithm *DSC* represent bottlenecks in the parallel execution flow because all processes must perform them. In particular, the bottleneck caused by the factorization 3^* of C adversely affects the parallel scaling of Algorithm *DSC*. This adverse behavior is caused by the fact that the Schur complement C is dense, and its factorization starts to dominate the overall execution time when the ratio of scenarios per process become small. We address this bottleneck by using iterative methods that make use of the preconditioner S_n introduced and analyzed in Section 3. The factorization of C is completely removed from the parallel execution flow, and the only bottleneck remains in the much less expensive solve phase.

In this new approach, a Krylov-type iterative solver is applied to the system $C\Delta u_0 = r_0 - \sum_{i=1}^{N} L_{Ni} w_i$, since the factors of C are not available anymore. In general, preconditioned Krylov subspace methods require the inverse of the preconditioner to be applied to a vector at each Krylov iteration. In our case, this step is done cheaply by means of triangular solves with factors of the preconditioner. The factors of the preconditioner are computed by a separate process $P+1$ *in the same time* the other P processes compute the terms of the Schur complement matrix C. Therefore, although the factorization of the preconditioner has the same cost as the factorization of the Schur complement, it does not create the factorization bottleneck of the Schur complement.

A slightly more sophisticated scenario scheduling mechanism than that of Algorithm *DSC* is implemented for Algorithm *PSC*. Namely, let $\mathcal{K} = \{k_1, k_2, \ldots, k_n\}$ be the scenarios based on which the preconditioner S_n is constructed, and let $\overline{\mathcal{N}} = \{1, 2, \ldots, N\} \setminus \mathcal{K}$ be the set of remaining scenarios. The scenarios for $\mathcal{K}$ are assigned to all $P+1$ processes, and the scenarios from $\overline{\mathcal{N}}$ are assigned only to processes 1, 2 through P. Once the scenarios from $\mathcal{K}$ are finished and S_n is computed, processes 1, 2 through P continue to compute the terms from the full Schur complement C, while process $P+1$ starts the factorization of the preconditioner. An even distribution of the scenarios is obtained by setting up and solving two distinct number partitioning problems for $\mathcal{K}$ and $\overline{\mathcal{N}}$ similar to the partitioning problem from Algorithm *DSC*. Let us denote the partitions found by $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_{P+1}$ for $\mathcal{K}$ and by $\overline{\mathcal{N}}_1, \overline{\mathcal{N}}_2, \ldots, \overline{\mathcal{N}}_P$ for $\overline{\mathcal{N}}$.

The complete set of operations and the interprocess communication patterns needed to solve the linear system $K\Delta u_0 = r$ are called generically Algorithm *PSC* (preconditioned Schur complement) and are presented separately in Algorithm *PSC-p* for processes $1, \ldots, P$ and in Algorithm *PSC-(P + 1)* for process $P+1$. The verb "reduce" in the listings of the algorithms refers to parallel computing operation that combines data held by different processes through an associative operator, in our case sumation, and accumulates the result on a single process (*reduce*) or on all processes (*all-reduce*). MPI routines *MPI_Reduce* and *MPI_Allreduce* correspond to these operations.

A comparison of Algorithm *PSC-p* and Algorithm *PSC-(P + 1)* shows that once the scenarios used in S_n are computed (steps 1.1 and 2.1), processes 1, 2 through P continue to compute the terms from the full Schur complement C (steps 1.2 and 2.2),

```
Algorithm PSC-p ($p = 1, 2, \dots, P$)
Factorization phase
1.1.  For each $i \in \mathcal{K}_p$ factorize $L_i D_i L_i^T = K_i$;
2.1.  Compute $C_{\mathcal{K}_p} = -\sum_{i\in\mathcal{K}_p} B_i^T K_i^{-1} B_i$, reduce $C_{\mathcal{K}} = \sum_{r=1}^{P+1} C_{\mathcal{K}_r}$ on
      process $P+1$ and on process 1;
1.2.  For each $i \in \overline{\mathcal{N}}_p$ factorize $L_i D_i L_i^T = K_i$;
2.2.  Compute $C_{\overline{\mathcal{N}}_p} = -\sum_{i\in\overline{\mathcal{N}}_p} B_i^T K_i^{-1} B_i$, reduce $C_{\overline{\mathcal{N}}} = \sum_{r=1}^{P} C_{\overline{\mathcal{N}}_r}$ on process 1, and,
      on process 1 only, compute $C = C_{\mathcal{K}} + C_{\overline{\mathcal{N}}} + K_0$;
Back substitution phase
4.    For each $i \in \mathcal{K}_p \cup \overline{\mathcal{N}}_p$ solve $w_i = L_i^{-1} r_i$;
5.    Compute $\sum_{i\in\mathcal{K}_p\cup\overline{\mathcal{N}}_p} L_{Ni} w_i$ and reduce $\sum_{i=1}^{N} L_{Ni} w_i$ on process 1;
Iterative preconditioned solve (6, 8, 9)
      Krylov iterative solve for $\Delta u_0 = C^{-1}(r_0 - \sum_{i=1}^{N} L_{Ni} w_i)$ (only process 1);
      Process 1 sends and processes $2, \dots, P$ receive $\Delta u_0$;
Diagonal solve phase
7.    For each $i \in \mathcal{K}_p \cup \overline{\mathcal{N}}_p$ solve $v_i = D_i^{-1} w_i$;
Forward substitution phase
10.   For each $i \in \mathcal{K}_p \cup \overline{\mathcal{N}}_p$ solve $\Delta u_i = L_i^{-T}(v_i - L_{Ni}\Delta u_0)$.
```

```
Algorithm PSC-(P+1)
factorization phase
1.1.  For each $i \in \mathcal{K}_{P+1}$ factorize $L_i D_i L_i^T = K_i$;
2.1.  Compute $C_{\mathcal{K}_{P+1}} = -\sum_{i\in\mathcal{K}_{P+1}} B_i^T K_i^{-1} B_i$, reduce $C_{\mathcal{K}} = \sum_{r=1}^{P+1} C_{\mathcal{K}_r}$ on
      process $P+1$ and on process 1, and compute $M = C_{\mathcal{K}} + K_0$;
3.    Factorize $L_M D_M L_M^T = M$;
Back substitution phase
4.    For each $i \in \mathcal{K}_{P+1}$ solve $w_i = L_i^{-1} r_i$;
5.    Compute $\sum_{i\in\mathcal{K}_{P+1}} L_{Ni} w_i$ and reduce $\sum_{i=1}^{N} L_{Ni} w_i$ on process 1;
Iterative preconditioned solve (6, 8, 9)
      Apply preconditioner $M^{-1}$ as many times process 1 needs it;
      Receive $\Delta u_0$ from process 1;
Diagonal solve phase
7.    For each $i \in \mathcal{K}_{P+1}$ solve $v_i = D_i^{-1} w_i$;
Forward substitution phase
10.   For each $i \in \mathcal{K}_{P+1}$ solve $\Delta u_i = L_i^{-T}(v_i - L_{Ni}\Delta u_0)$.
```

while process $P+1$ starts the factorization of the preconditioner (step 3). The largest number n of scenarios included in S_n is chosen such that process $P+1$ completes step 3 before the other P processes finish steps 1.2 and 2.2. Therefore, the bottleneck 3* from Algorithm *DSC* is removed from the parallel execution flow of Algorithm *PSC*.

On the other hand, the iterative preconditioned Krylov solve of Algorithm *PSC* is slightly more expensive than the corresponding solve with the factors of the Schur complement (steps 6*, 8*, and 9*) of Algorithm *DSC*. More exactly, each Krylov iteration requires the action of the inverse of the preconditioner, which has the same cost as solving with the factors of the Schur complement, namely, $O\left((n_{x_0}+m_{x_0})^2\right)$, and a total overhead of $O\left(l(n_{x_0}+m_{x_0})^2\right)$ is present in the iterative solve phase of Algorithm *PSC*, where l is the number of Krylov iterations. However, the savings from the removal of the direct factorization of the Schur matrix are $O\left((n_{x_0}+m_{x_0})^3\right)$, therefore, Algorithm *PSC* is more scalable than Algorithm *DSC* whenever $l << n_{x_0}+m_{x_0}$. In the numerical experiments presented in Section 5, the number l of iterations is no more than 400, usually less than 10, while $n_{x_0}+m_{x_0}$ is more than $10,000$.

The use of PSC in a single precessor environment is not recommended. In this situation, the factorization of the preconditioner can not be done in the same time with the calculations related to second-stage, and adds to total execution time. Therefore the most substantial benefit of PSC, *i.e.* the obliteration of the Schur complement factorization, is annihilated by the same-cost factorization of the preconditioner. Furthermore, the fact that the Krylov solve phase of PSC is several times more expensive than the corresponding solve phase of DSC causes PSC to be slower than DSC in any single process environment.

We have implemented a mechanism that decides at runtime whether to use PSC or DSC. This mechanism requires no user intervention, estimates the execution times needed by each of the two methods at the next IPM iteration and uses the one with shorter execution time. The estimates are obtained by timing the linear algebra of the Schur complement matrix (or the preconditioner) and the second-stage subproblems at the IPM iteration that just ended. Such estimates are very reliable in the context of IPMs, staying constant for the most of the iterates, and some of them steadily increasing as the iterates approach the solution (because of the ill-conditioning of IPMs). The implementation uses DSC for the first iteration, unless instructed otherwise, and assumes that PSC needs $l=10$ Krylov iterations when $\mu>0.1$ and $l=10\,\text{floor}(\log_{10}(1/\mu))$ otherwise. Our tests showed that this decision mechanism is robust.

4.2 Implementation of the Krylov subspace iterative methods

The iterative methods we have implemented are BiCGStab [30] and the preconditioned projected conjugate gradient (PPCG) from [29]. We apply BiCGStab to the saddle-point linear system involving the Schur complement matrix C and use the constraint preconditioner M given by (34). The products involving C are computed on process 1, and the products involving M^{-1} are performed remotely on process $P+1$. At each iteration two of such products are needed.

The PPCG method solves saddle-point linear system

$$\begin{bmatrix} S_N & T_0^T \\ T_0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix},$$

by making use of a basis Z_0 that spans the null space of T_0. The PPCG algorithm starts at a point x that satisfies $T_0x=r_2$ and updates x by using a direction that lies in the range of Z_0 (*i.e.,* the null space of T_0) and computed by means of the (preconditioned) projector operator

$$P=Z_0(Z_0^TS_n^{-1}Z_0)Z_0^T.$$

Consequently, the iterates x remain in the affine subspace given by $T_0x=r_2$. Once the iterate x converges, y is found from $(T_0T_0^T)y=r_1-T_0S_Nx$. The cost of solving for y is small since the factorization is done only once, at the first IPM iteration. An important observation here is that Z_0 does not have to be explicitly computed. Instead, the projection $v = Pu$ can be performed efficiently by solving the linear system

$$\begin{bmatrix} S_n & T_0^T \\ T_0 & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix},$$

that is, by solving with the factors of the *constraint preconditioner* M, not with those of the stochastic preconditioner S_n. The residual update strategy [29] enforced to reduce the effects of roundoff errors occurring in the computation of the projected residual requires an extra solve with the factors of M at each iteration of the PPCG. Therefore, two applications of the inverse of the preconditioner is needed at each PPCG iteration, which makes PPCG and BiCGStab to have about the same cost per one iteration.

5 Computational results

In this section we present the computational results of the DSC and PSC approaches presented in the previous section. We have used *Fusion*, a 320-node computing cluster at Argonne National Laboratory using an InfiniBand QDR interconnect with 2 μsec latency. Each node is dual-socket, quad-core and therefore has 8 cores. Each core operates at 2.53 GHz. A minimum of 16 GB memory is available for each node.

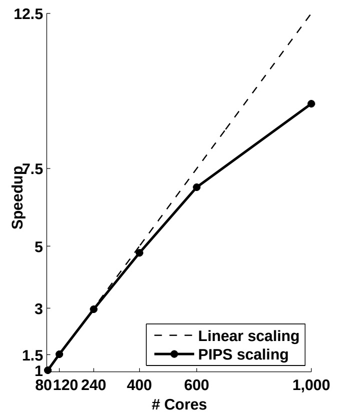

Fig. 1 Speedup of DSC method in solving a unit commitment problem with a relatively small first stage.

Both the DSC and PSC algorithms were implemented in a parallel interior-point solver for stochastic programming (PIPS) we developed recently. PIPS uses the object-oriented design of OOQP [21] and reuses many of OOQP's classes. The largest problem we have solved with PIPS consists of 28.9 millions variables: $8,892$ first-stage variable, $7,226$ second-stage variables and $4,000$ scenarios. The problem is similar to the unit commitment problem described in Section 5.1 but has simplified dynamics and has been replicated artificially for testing purposes. The speedup obtained by PIPS using the DSC approach is shown in Figure 1. Parallel efficiencies of 0.95, 0.92, and 0.77 are reached when the number of cores is increased from 80 to 400, 600, and $1,000$, respectively. In the case of a building energy system control problem [54], almost perfect scaling was obtained from 10 cores to 50 cores. The ratio of the number of first-stage variables and the number of second-stage variables is small for both problems, and therefore the bottleneck created by the Schur complement does not affect significantly the scalability. However, this is not the case for the problems described in the following section, and the PSC method is able to overcome this limiting behavior as will be shown in Section 5.3.

5.1 The test problems

The computational framework presented in [14] integrates the Weather Research and Forecasting (WRF) model in stochastic unit commitment and energy dispatch formulations that account for the wind power in the generation of the electricity. The unit commitment (UC) problem of the above study is a mixed integer linear programming problem and involves a network of 10 thermal generators and 12 wind farms (each of 50 turbines). We solved a relaxation of the problem and obtained the sample average approximation problem by using a set of 30 scenarios for the wind power levels. We denote this instance as *UC1*. A second instance, *UC1Q* was artificially obtained by considering random quadratic costs occurring in the operation of the thermal units. *UC1* and *UC1Q* are used in the next subsection to study the spectrum of the preconditioner.

Two larger UC problems, denoted by *UC2* and *UC2Q*, were obtained by replicating three times the generators of *UC1* and *UC1Q*, respectively. A set of 120 (resampled) scenarios were used for these instances. The problems have a total of $210,000$ variables and $443,000$ constraints. The first-stage problem is relatively large when compared to the second-stage problems, with $10,800$ variables and $24,000$ constraints and $1,656$ variables and $3,494$ constraints, respectively. *UC2* and *UC2Q* problems are used in subsection 5.3 to investigate and compare the scalability of the DSC and PSC methods.

While *UC1* and *UC2* are linear programming problems ($Q_0 = 0$ and $Q(\omega) = 0$ in (2)), *UC1Q* and *UC2Q* are convex quadratic problems with Q_0 and $Q(\omega)$ being semidefinite diagonal matrices. We note that the technology matrix T and the recourse matrix W are fixed (deterministic) for all instances. The randomness occurs only in the right-hand side $b(\omega)$ for the linear problems and in the right-hand side and quadratic coefficients $Q(\omega)$ for the quadratic problems.

5.2 Quality of preconditioners

The first numerical experiment investigates the quality of the S_n preconditioner when *UC1* and *UC1Q* instances are solved. The number of scenarios included in S_n was $n = 5$ out of a total of 30 scenarios. We first look at the eigenvalues of the preconditioned matrix $S_n^{-1}S_N$ at different moments in the solving process: IPM iterations 5, 15, 25, and 35. The smallest and largest 50 eigenvalues of $S_n^{-1}S_N$ are displayed in the left plots of Figure 2. This figure shows a solid clustering of the eigenvalues at 1. The dispersion of the extreme several eigenvalues as the optimization approaches the solution is caused by the adverse effect of the ill-conditioning of the diagonals D_i, $i = 0, \ldots, n$ on the average approximation S_n of S_N. *UC1Q* exhibits a more severe dispersion of the eigenvalues than the *UC1* instance towards the end of the optimization. This behavior is likely caused by the presence of the randomness in the objective of *UC1Q* and reflects in a increased number the Krylov iterations for *UC1Q* over *UC1* close to optimality.

Both BiCGStab and PPCG take just less than 3 iterations for more than half of the outer iterations but need an increasing number of iterations after that. For the *UC1* instance we recorded a maximum of 88 BiCGStab iterations and 64 PPCG iterations at the last (36th) IPM iteration. As a comparison, at the same IPM iteration for *UC1Q*, 186 BiCGStab iterations and 132 PPCG iterations were obtained . In the case of *UC1Q*, there was an extra IPM iteration at which a maximum of 308 BiCGStab iterations and 200 PPCG iterations were recorded.

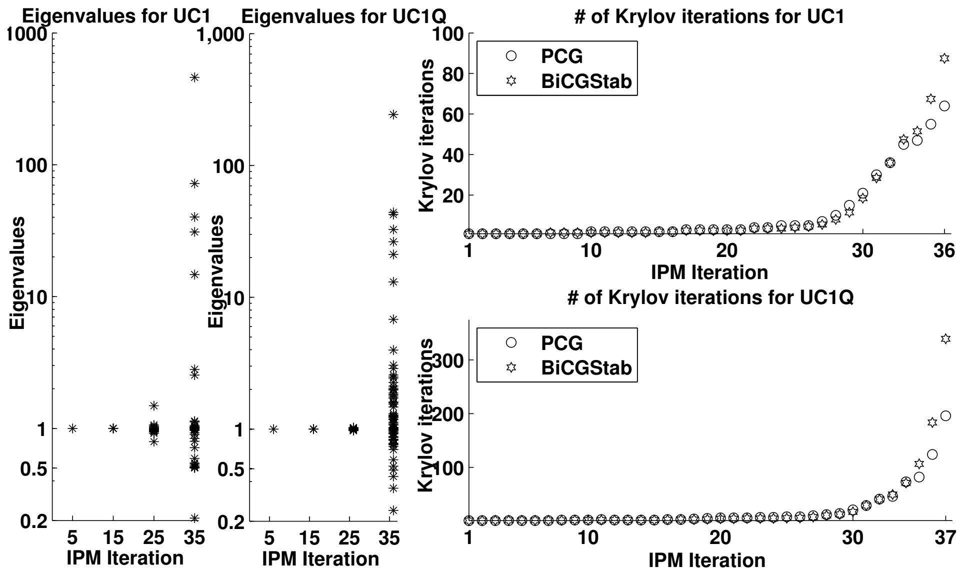

Fig. 2 In the left figures, the 50 largest and the 50 smallest eigenvalues of the preconditioned matrix at IPM iterations 5, 15, 25, and 35 are depicted for *UC1* and *UC1Q*. The right figures show the number of inner iterations needed to solve the predictor system at each IPM iteration for the same two UC instances.

The linear systems are 2608×2608 and extremely ill-conditioned at the last IPM iterations (condition number is $O(10^{13})$, see Table 1). They are solved with a relative error of 10^{-8} for all IPM iterations. The relative error is the ratio of the norm of residual and the norm of the right-hand side. We mention that the ill-conditioning occurring during the last IPM iterations can be ameliorated and the number of Krylov iterations can be potentially reduced by using techniques for early detection and removal of the inactive first-stage variables and inequality constraints [13, 27] or by regularizing the saddle-point system [1].

IPM	*UC1* instance		*UC1Q* instance	
Iteration	$\kappa(C)$	$\kappa(M^{-1}C)$	$\kappa(C)$	$\kappa(M^{-1}C)$
1	$7.68 \cdot 10^{8}$	1.04	$3.27 \cdot 10^{8}$	1.08
5	$1.10 \cdot 10^{12}$	1.11	$1.46 \cdot 10^{8}$	1.24
15	$4.69 \cdot 10^{13}$	$6.51 \cdot 10^{4}$	$3.70 \cdot 10^{13}$	$5.99 \cdot 10^{4}$
25	$2.18 \cdot 10^{10}$	$4.42 \cdot 10^{6}$	$7.86 \cdot 10^{10}$	$5.28 \cdot 10^{3}$
35	$1.82 \cdot 10^{13}$	$1.82 \cdot 10^{10}$	$8.44 \cdot 10^{13}$	$1.19 \cdot 10^{10}$

Table 1 Condition numbers of the Schur complement matrix C and of the preconditioned matrix $M^{-1}C$. M is constraint preconditioner given by (34).

Another important observation is that the number of iterations needed by BiCGStab and PPCG is almost the same for the first 30 IPM iterations. After that PPCG takes advantage of the positive-definiteness of the $(1,1)$ block and needs fewer iterations than BiCGStab to solve the system within the same accuracy.

5.3 Practical performance of the preconditioners

We use the *UC2* and *UC2Q* problems to study the scalability of DSC and PSC. The mechanism described in Section 4.1 that decides whether to use DSC or PSC was disabled for the tests presented here. *Strong scaling* is investigated, that is, the same problem was solved with an increasing number of cores. A linear scaling occurs when the execution time decreases linearly as the number of cores increases.

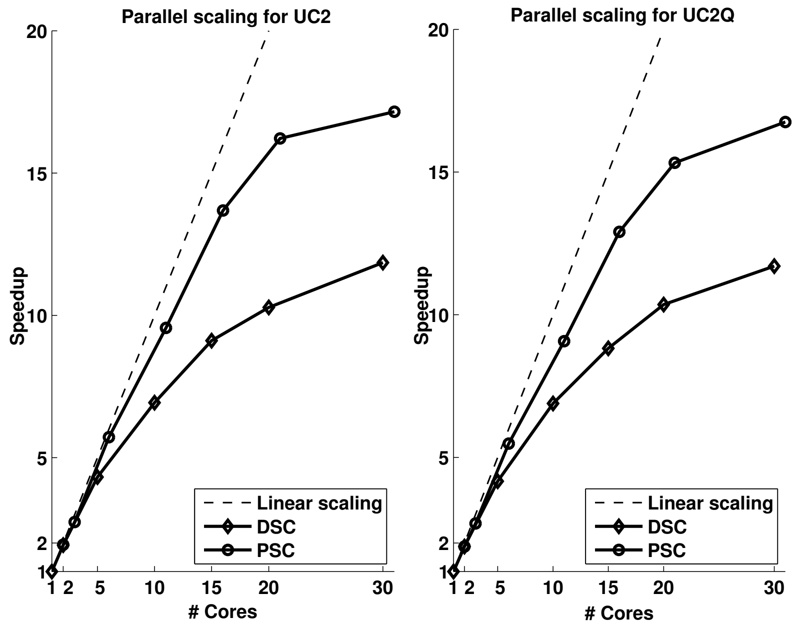

Fig. 3 Strong scaling of DSC and PSC using P and $P+1$ cores, respectively ($P = \{1, 2, 5, 10, 15, 20, 30\}$). Left plot obtained for *UC2*, right plot for *UC2Q*.

For the strong scaling investigations of this section we solved *UC2* and *UC2Q* problems using DSC and PSC with $P = \{1, 2, 5, 10, 15, 20, 30\}$ cores. Both problems have a very large Schur complement matrix which adversely affects the strong scaling of the DSC, as can be seen in Figure 3. PSC does not exhibit the same limitation and scales linearly for a wide range of cores, 2-20. When using more than 20 cores, however, the factorization of the preconditioner does not finish before the scenarios are done and thus creates its own bottleneck in the parallel execution flow. Consequently, the scaling is no longer linear, as shown in Figure 3.

We have used DSC with 1 core as a reference for the strong scaling of not only DSC but also PSC. Therefore, Figure 3 also shows which method is faster. For example, the reduction in the solve time obtained by PSC over DSC for *UC2* is 20%, 28%, 33%, and 24% for 10, 15, 20, and 30 cores, respectively. PSC yields similar gains for *UC2Q* instance.

6 Conclusions

In this paper, we presented a preconditioning technique for Schur complement linear systems arising in the interior-point based parallelization of the two-stage stochastic programming problems with recourse. The spectral analysis based on the Hoeffding's inequality from the theory of large deviations showed that an exponentially fast clustering of the eigenvalues occurs with the number of scenarios incorporated in the preconditioner. The good quality of the preconditioner allows the IPM saddle-point systems to be solved with the same accuracy as the direct solvers with a small number of Krylov iterations.

Numerical experiments involving large instances of a stochastic unit commitment problem indicate good practical performance. The small number of Krylov iterations needed to solve the (preconditioned) Schur complement linear systems significantly reduces the bottleneck caused by the direct factorization of the Schur complement matrix. Consequently, the preconditioning technique exhibits a linear strong scaling for a reasonably large number of processes. This range can be further extended by preconditioning with more than one process. This can be done for example by using a distributed-memory parallel dense linear solver as Elemental [46] or ScaLapack [9] for the factorization and solves with the preconditioner. We plan to follow this approach and to implement it in PIPS.

The preconditioning algorithm PSC can also be used when solving an SAA T-stage stochastic programming, $T > 2$. In this case, the IPM saddle-point linear systems have an arrow shape *nested* structure; that is, each of the diagonal blocks K_i, $i = 1, \ldots, N$, of K from (11) have the arrow shape of K, recursively defined for $T > 3$. The Schur complement technique we described in this paper can be then used for each of the first $T - 1$ stages to decompose the problem [24]. The scalability of the DSC method may be affected by stages with a disproportionately large number of variables, and the PSC method should be used for such stages. We also note that a mixed DSC-PSC approach for multistage problems is robust, in the sense that the implementation can decide at runtime for each stage whether it is profitable or not to employ the preconditioning based on the decision mechanism presented in Section 4.1.

Acknowledgments

The authors are grateful to John Birge for discussions on the issues in the manuscript, and to three referees whose constructive comments helped to improve the paper. This work was supported by the U.S. Department of Energy through contract DE-AC02-06CH11357.

References

1. Altman, A., Gondzio, J.: Regularized symmetric indefinite systems in interior-point methods for linear and quadratic optimization. Optimization Methods and Software **11**(1-4), 275–302 (1999)
2. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. ACTA NUMERICA **14**, 1–137 (2005)
3. Bergamaschi, L., Gondzio, J., Zilli, G.: Preconditioning indefinite systems in interior point methods for optimization. Computational Optimization and Applications **28**(2), 149–171 (2004)

4. Birge, J.R.: Current trends in stochastic programming computation and applications. Tech. rep., Department of Industrial and Operations Engineering, University of Michigan, Ann Harbour, Michigan (1995)
5. Birge, J.R., Chen, X., Qi, L.: A stochastic Newton method for stochastic quadratic programs with recourse. Tech. rep., Applied Mathematics Preprint AM94/9, School of Mathematics, the University of New South Wales (1995)
6. Birge, J.R., Holmes, D.F.: Efficient solution of two stage stochastic linear programs using interior point methods. Computational Optimization and Applications **1**, 245–276 (1992)
7. Birge, J.R., Louveaux, F.: Introduction to stochastic programming. Springer-Verlag, New York (1997)
8. Birge, J.R., Qi, L.: Computing block-angular Karmarkar projections with applications to stochastic programming. Manage. Sci. **34**(12), 1472–1479 (1988)
9. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA, (1997)
10. Bunch, J.R., Kaufman, L.: Some stable methods for calculating inertia and solving symmetric linear systems. Mathematics of Computation **31**(137), 163–179 (1977)
11. Bunch, J.R., Parlett, B.N.: Direct methods for solving symmetric indefinite systems of linear equations. SIAM Journal on Numerical Analysis **8**(4), 639–655 (1971)
12. Cafieri, S., DApuzzo, M., Marino, M., Mucherino, A., Toraldo, G.: Interior-point solver for large-scale quadratic programming problems with bound constraints. Journal of Optimization Theory and Applications **129**, 55–75 (2006)
13. Castro, J.: A specialized interior-point algorithm for multicommodity network flows. SIAM Journal on Optimization **10**(3), 852–877 (2000)
14. Constantinescu, E.M., Zavala, V.M., Rocklin, M., Lee, S., Anitescu, M.: A computational framework for uncertainty quantification and stochastic optimization in unit commitment with wind power generation. IEEE Transactions on Power Systems, to appear (2010)
15. Czyzyk, J., Mehrotra, S., Wright, S.J.: PCx user guide. Tech. Rep. OTC 96/01, Optimization Technology Center, Argonne National Laboratory and Northwestern University (1996)
16. Dantzig, G.B., Infanger, G.: Large-scale stochastic linear programs – Importance sampling and Benders decomposition. In: Computational and Applied Mathematics, I, pp. 111–120. North-Holland, Amsterdam (1992)
17. D'Apuzzo, M., Simone, V., Serafino, D.: On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods. Comput. Optim. Appl. **45**, 283–310 (2010)
18. Dollar, H.S., Gould, N.I.M., Schilders, W.H.A., Wathen, A.J.: Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems. SIAM Journal on Matrix Analysis and Applications **28**(1), 170–189 (2006)
19. Dollar, H.S., Wathen, A.J.: Approximate factorization constraint preconditioners for saddle-point matrices. SIAM Journal on Scientific Computing **27**(5), 1555–1572 (2006)
20. Ermoliev, Y.M.: Stochastic quasigradient methods. In: Numerical techniques for stochastic optimization, *Springer Ser. Comput. Math.*, vol. 10, pp. 141–185. Springer, Berlin (1988)
21. Gertz, E.M., Wright, S.J.: Object-oriented software for quadratic programming. ACM Transactions on Mathematical Software **29**(1), 58–81 (2003)
22. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. The Johns Hopkins University Press (1996)
23. Gondzio, J.: HOPDM (version 2.12) - a fast LP solver based on a primal-dual interior point method. European Journal of Operational Research **85**, 221–225 (1995)
24. Gondzio, J., Grothey, A.: Direct solution of linear systems of size 10^9 arising in optimization with interior point methods. In: PPAM, pp. 513–525 (2005)
25. Gondzio, J., Grothey, A.: Parallel interior-point solver for structured quadratic programs: Application to financial planning problems. Annals of Operations Research **152**(1), 319–339 (2007)
26. Gondzio, J., Grothey, A.: Exploiting structure in parallel implementation of interior point methods for optimization. Computational Management Science **6**(2), 135–160 (2009)
27. Gondzio, J., Makowski, M.: Solving a class of LP problems with a primal–dual logarithmic barrier method. European Journal of Operational Research **80**, 184–192 (1995)
28. Gondzio, J., Sarkissian, R.: Parallel interior point solver for structured linear programs. Mathematical Programming **96**, 561–584 (2000)

29. Gould, N.I.M., Hribar, M.E., Nocedal, J.: On the solution of equality constrained quadratic programming problems arising in optimization. SIAM Journal on Scientific Computing **23**(4), 1376–1395 (2001)
30. Greenbaum, A.: Iterative Methods for Solving Linear Systems. SIAM Series on Frontiers in Applied Mathematics, Philadelphia, PA (1997)
31. Güler, O.: Existence of interior points and interior paths in nonlinear monotone complementarity problems. Math. Oper. Res. **18**(1), 128–147 (1993)
32. Higle, J.L., Sen, S.: Stochastic decomposition: An algorithm for two-stage linear programs with recourse. Mathematics of Operations Research **16**, 650–669 (1991)
33. Keller, C., Gould, N.I.M., Wathen, A.J.: Constraint preconditioning for indefinite linear systems. SIAM J. Matrix Anal. Appl. **21**(4), 1300–1317 (2000)
34. King, A.J.: An implementation of the Lagrangian finite-generation method. In: Numerical techniques for stochastic optimization, *Springer Ser. Comput. Math.*, vol. 10, pp. 295–311. Springer, Berlin (1988)
35. Linderoth, J., Wright, S.J.: Decomposition algorithms for stochastic programming on a computational grid. Comput. Optim. Appl. **24**(2-3), 207–250 (2003)
36. Lukšan, L., Vlček, J.: Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems. Numerical Linear Algebra with Applications **5**(3), 219–247 (1998)
37. Lustig, I.J., Marsten, R.E., Shanno, D.F.: On implementing Mehrotra's predictor–corrector interior-point method for linear programming. SIAM Journal on Optimization **2**(3), 435–449 (1992)
38. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**, 239–245 (1979)
39. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM Journal on Optimization **2**(4), 575–601 (1992)
40. Mehrotra, S., Ozevin, M.G.: Decomposition-based interior point methods for two-stage stochastic semidefinite programming. SIAM J. on Optimization **18**(1), 206–222 (2007)
41. Mehrotra, S., Ozevin, M.G.: Decomposition based interior point methods for two-stage stochastic convex quadratic programs with recourse. Oper. Res. **57**(4), 964–974 (2009)
42. Mehrotra, S., Ozevin, M.G.: On the implementation of interior point decomposition algorithms for two-stage stochastic conic programs. SIAM Journal on Optimization **19**(4), 1846–1880 (2009)
43. Monteiro, R.D.C., Pang, J.S.: Properties of an interior-point mapping for mixed complementarity problems. Mathematics of Operations Research **21**(3), 629–654 (1996)
44. Murphy, M.F., Golub, G.H., Wathen, A.J.: A note on preconditioning for indefinite linear systems. SIAM Journal on Scientific Computing **21**(6), 1969–1972 (2000)
45. Pflug, G.C., Halada, L.: A note on the recursive and parallel structure of the Birge and Qi factorization for tree structured linear programs. Comput. Optim. Appl. **24**(2-3), 251–265 (2003)
46. Poulson, J., Marker, B., van de Geijn, R.A.: Elemental: A new framework for distributed memory dense matrix computations (flame working note #44). Tech. rep., Institute for Computational Engineering and Sciences, The University of Texas at Austin (2010)
47. Rockafellar, R.T., Wets, R.J.B.: A Lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming. Math. Programming Stud. **28**, 63–93 (1986). Stochastic programming 84. II
48. Rockafellar, R.T., Wets, R.J.B.: Scenarios and policy aggregation in optimization under uncertainty. Mathematics of Operations Research **16**(1), 119–147 (1991)
49. Rosa, C.H., Ruszczyński, A.: On augmented Lagrangian decomposition methods for multistage stochastic programs. Ann. Oper. Res. **64**, 289–309 (1996). Stochastic programming, algorithms and models (Lillehammer, 1994)
50. Ruszczyński, A.: A regularized decomposition method for minimizing a sum of polyhedral functions. Mathematical Programming **35**, 309–333 (1986)
51. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on Stochastic Programming: Modeling and Theory. MPS/SIAM Series on Optimization 9, Philadelphia, PA (2009)
52. Van Slyke, R., Wets, R.J.: L-shaped linear programs with applications to control and stochastic programming. SIAM Journal on Applied Mathematics **17**, 638–663 (1969)
53. Wright, S.J.: Primal-Dual Interior-Point Methods. Society for Industrial and Applied Mathematics, Philadelphia, PA (1997)

54. Zavala, V.M., Constantinescu, E.M., Krause, T., Anitescu, M.: On-line economic optimization of energy systems using weather forecast information. Journal of Process Control **19**, 1725–1736 (2009)
55. Zavala, V.M., Laird, C.D., Biegler, L.T.: Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. Chemical Engineering Science **63**(19), 4834–4845 (2008)
56. Zhang, D., Zhang, Y.: A Mehrotra-type predictor-corrector algorithm with polynomiality and Q-subquadratic convergence. Ann. Oper. Res. **62**, 131–150 (1996)
57. Zhang, Y.: Solving large-scale linear programs by interior-point methods under the Matlab environment. Tech. Rep. TR96-01, University of Maryland Baltimore County (1996)
58. Zhao, G.: A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs. Mathematical Programming **90**(3), 507–536 (2001)